CPP05 – EX01 : FORM UP, MAGGOTS!

Soutenance orale – Document pédagogique complet

Ce document t'aide à présenter l'exercice ex01 à l'oral, comme si tu l'expliquais à un début débutant. Chaque section contient les notions essentielles et des exemples concrets.

1. Introduction à l'exercice

Dans l'ex00, nous avons créé la classe Bureaucrat avec un nom et un grade. Dans l'ex01, nous introduisons une nouvelle classe : Form. L'objectif est de permettre à un bureaucrate de tenter de signer un formulaire, en respectant des règles strictes sur les grades et les exceptions. Cet exercice enseigne la relation entre objets, la validation et les exceptions, piliers de l'OOP.

2. La classe Form

Un Form représente un document administratif à signer. Il contient :

- un nom constant,

- un booléen indiquant s'il est signé,

- un grade minimal pour le signer,

- un grade minimal pour l'exécuter.

Les grades doivent être entre 1 (meilleur) et 150 (pire).

Cas concret (valide)

Form f("TopSecret", 20, 5);

→ Le formulaire est valide et non signé.

Cas contraire (invalide)

Form bad("Impossible", 0, 10);

→ Grade trop haut : Form::GradeTooHighException.

3. La méthode beSigned()

Le formulaire peut être signé si le grade du bureaucrate est suffisamment bon :
bureaucrat.getGrade() <= gradeToSign.
Sinon, on lance Form::GradeTooLowException.

Cas concret (signature réussie)

Alice (grade 10) signe TopSecret (grade requis : 20).
→ Succès, _isSigned passe à true.

Cas contraire (signature échouée)

Bob (grade 150) tente de signer ImportantPaper (grade requis : 100).
→ Exception : Form::GradeTooLowException.

4. Bureaucrat::signForm()

Cette méthode tente de signer le formulaire via form.beSigned(*this).
Elle utilise un bloc try/catch pour afficher :
- signed si succès,
- couldn’t sign because si échec.
C’est exactement ce que demande le sujet.

5. Surcharge de l’opérateur <<

Nous surchargeons operator<< pour afficher toutes les informations importantes du formulaire :
nom, signed, gradeToSign, gradeToExecute.

6. Exceptions

Les exceptions permettent d'empêcher la création ou l'utilisation d'objets invalides. Elles assurent une exécution propre et contrôlée.

Dans cet exercice :

- GradeTooHighException signifie : grade < 1,

- GradeTooLowException signifie : grade > 150.

7. Respect des règles du module C++98

Ce projet respecte :

- les noms en UpperCamelCase,

- l'Orthodox Canonical Form,

- l'absence de STL avant le module 08,

- aucun using namespace std,

- compatibilité C++98.

8. Conclusion pour la soutenance

Cet exercice montre que tu sais :

- créer une classe respectant des contraintes strictes,

- faire collaborer deux classes entre elles,

- gérer proprement les erreurs via des exceptions,

- structurer un code lisible en C++98.

Ta compréhension doit surtout montrer que tu maîtrises la logique de validation des grades et la communication entre Bureaucrat et Form.